



ASP.NET Core 框架揭秘

Inside ASP.NET Core Framework

蒋金楠

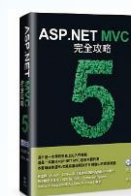
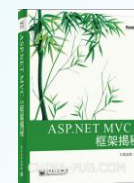
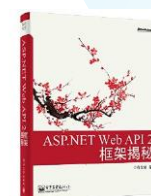
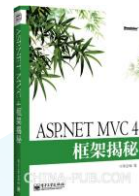
个人简介 - 蒋金楠



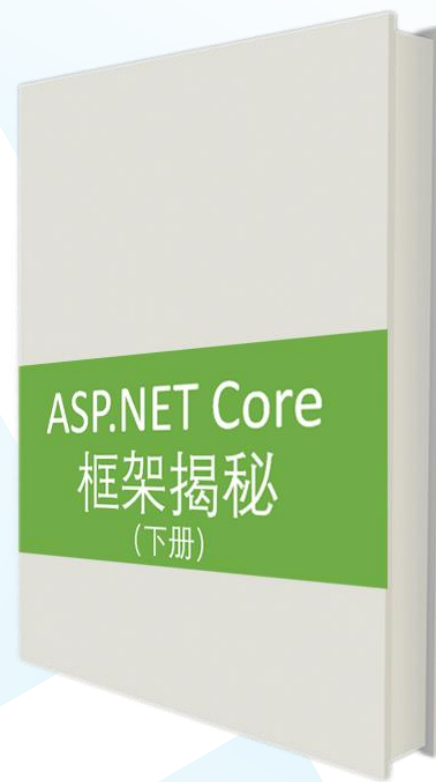
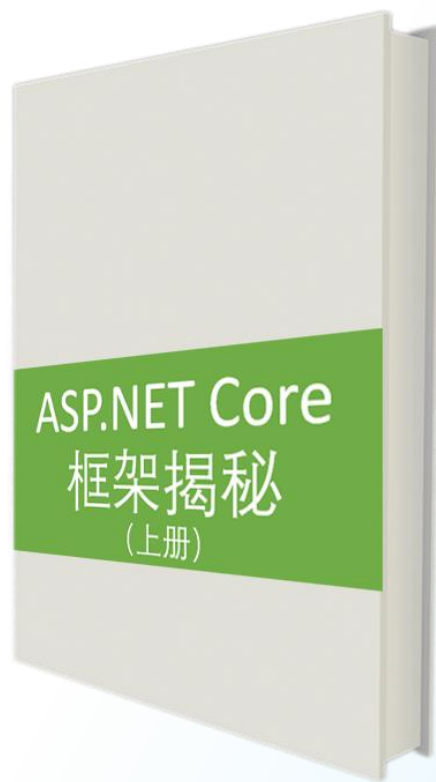
大内老A



X 12



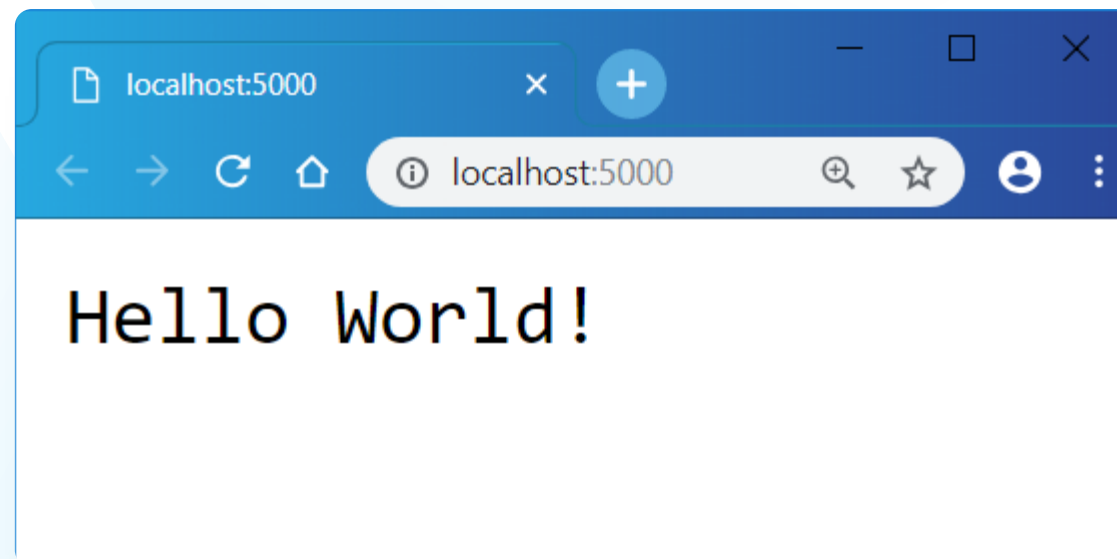
即将推出...



Hello World



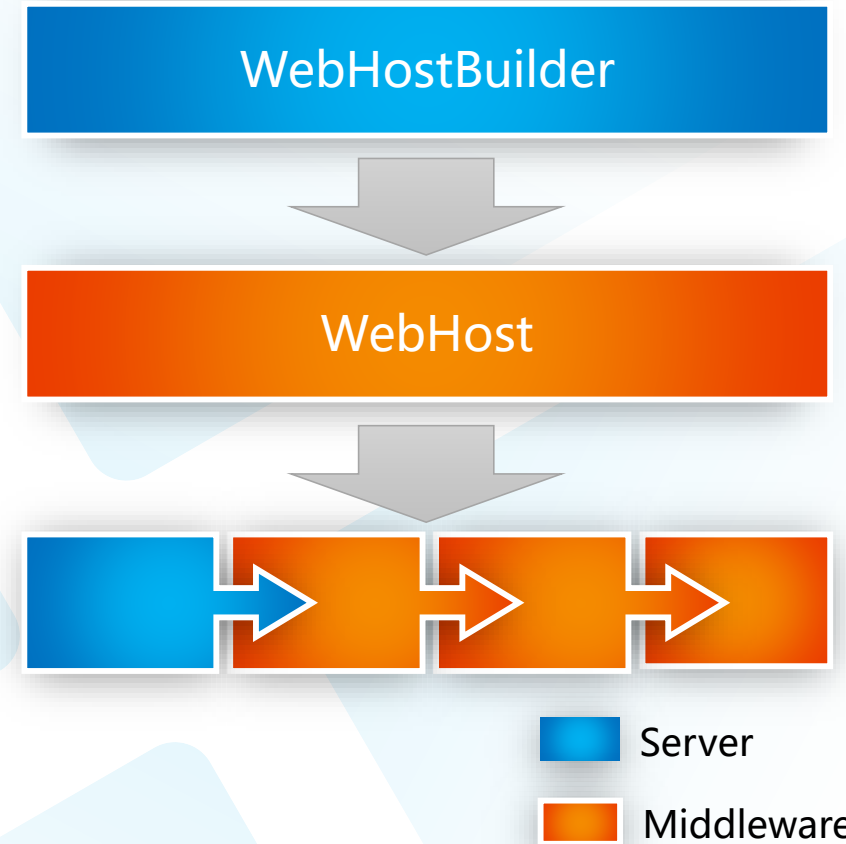
```
public class Program
{
    public static void Main()
    => new WebHostBuilder()
        .UseKestrel()
        .Configure(app => app.Run(context => context
            .Response.WriteAsync("Hello World!")))
        .Build()
        .Run();
}
```



Pipeline



```
public class Program
{
    public static void Main()
    => new WebHostBuilder()
        .UseKestrel()
        .Configure(app => app.Run(context => context
            .Response.WriteAsync("Hello World!")))
        .Build()
        .Run();
}
```





ASP.NET Core Mini



```
public class Program
{
    public static async Task Main()
    {
        await new WebHostBuilder()
            .UseHttpListener()
            .Configure(app => app
                .Use(FooMiddleware)
                .Use(BarMiddleware)
                .Use(BazMiddleware))
            .Build()
            .StartAsync();
    }
}
```

```
public class Program
{
    public static RequestDelegate FooMiddleware(RequestDelegate next)
    => async context => {
        await context.Response.WriteAsync("Foo=>");
        await next(context);
    };

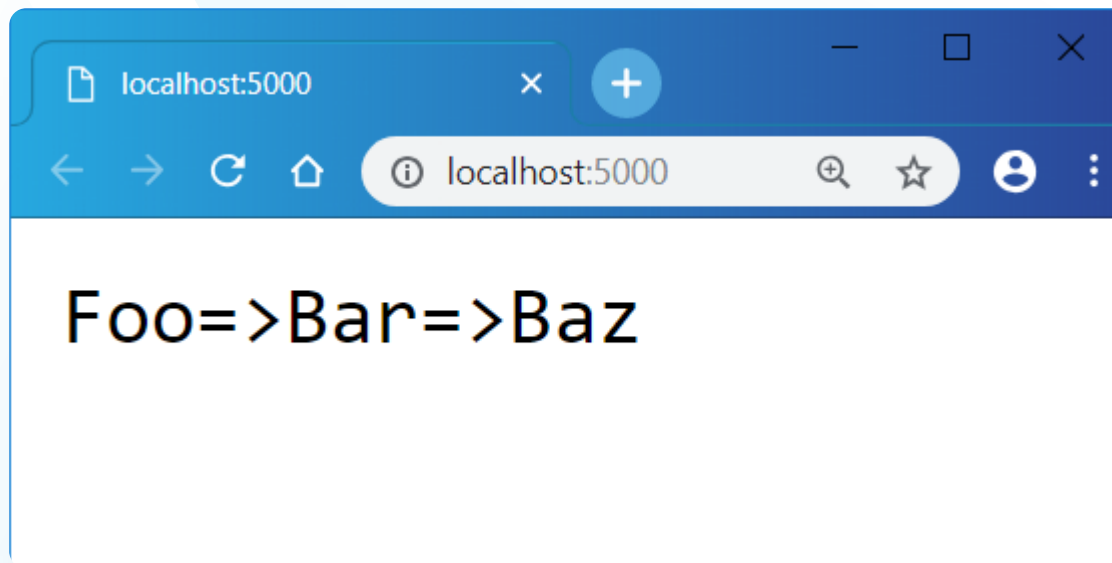
    public static RequestDelegate BarMiddleware(RequestDelegate next)
    => async context => {
        await context.Response.WriteAsync("Bar=>");
        await next(context);
    };

    public static RequestDelegate BazMiddleware(RequestDelegate next)
    => context => context.Response.WriteAsync("Baz");
}
```

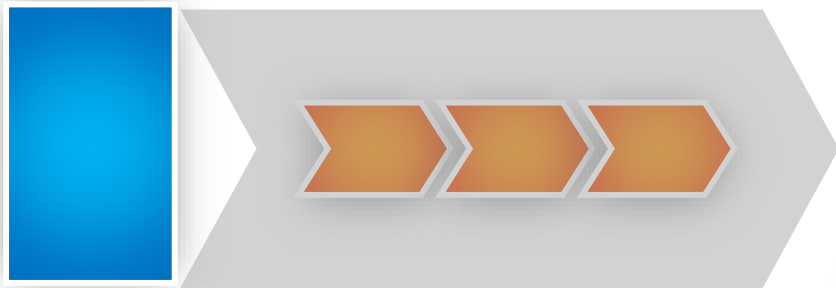
ASP.NET Core Mini




```
public class Program
{
    public static async Task Main()
    {
        await new WebHostBuilder()
            .UseHttpListener()
            .Configure(app => app
                .Use(FooMiddleware)
                .Use(BarMiddleware)
                .Use(BazMiddleware))
            .Build()
            .StartAsync();
    }
}
```



HttpContext



 Server

 Middleware

 HttpContext

```
public class HttpContext
{
    public HttpRequest Request { get; }
    public HttpResponse Response { get; }
}

public class HttpRequest
{
    public Uri Url { get; }
    public NameValueCollection Headers { get; }
    public Stream Body { get; }
}

public class HttpResponse
{
    public NameValueCollection Headers { get; }
    public Stream Body { get; }
    public int StatusCode { get; set; }
}
```

RequestDelegate



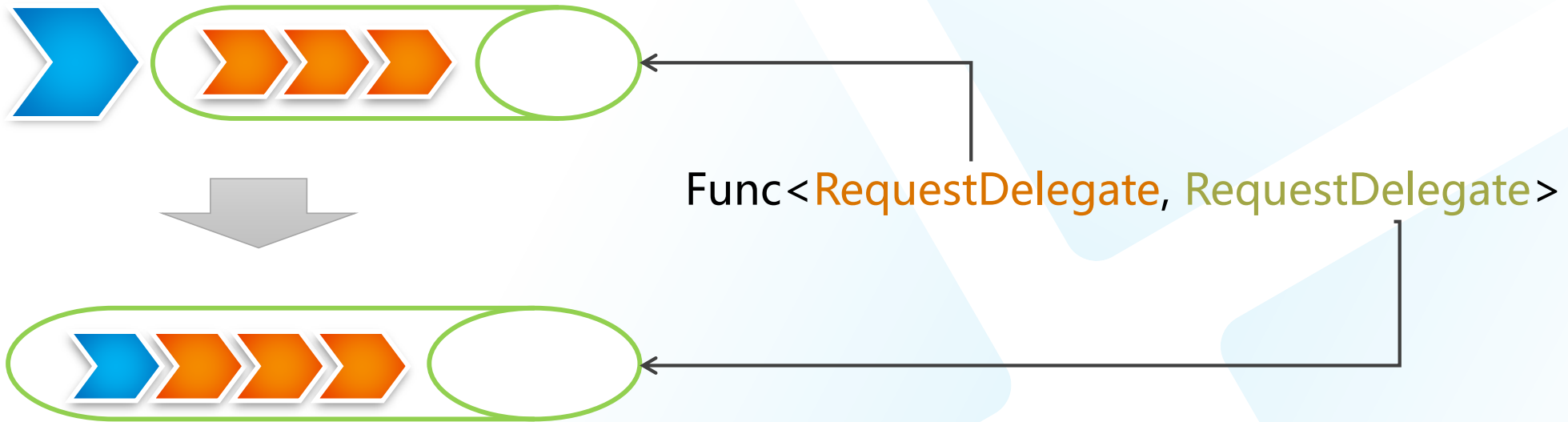
Server + Middlewares



Server + HttpHandler



Middleware



ApplicationBuilder



```
public interface IApplicationBuilder
{
    IApplicationBuilder Use(
        Func<RequestDelegate, RequestDelegate> middleware);
    RequestDelegate Build();
}
```

ApplicationBuilder



```
public class ApplicationBuilder : IApplicationBuilder
{
    private readonly List<Func<RequestDelegate, RequestDelegate>> _middlewares
        = new List<Func<RequestDelegate, RequestDelegate>>();
    public RequestDelegate Build() => httpContext =>
    {
        RequestDelegate next = _ => Task.Run(() => _.Response.StatusCode = 404);
        _middlewares.Reverse();
        foreach (var middleware in _middlewares)
        {
            next = middleware(next);
        }
        return next(httpContext);
    };
    public IApplicationBuilder Use(Func<RequestDelegate, RequestDelegate> middleware)
    {
        _middlewares.Add(middleware);
        return this;
    }
}
```

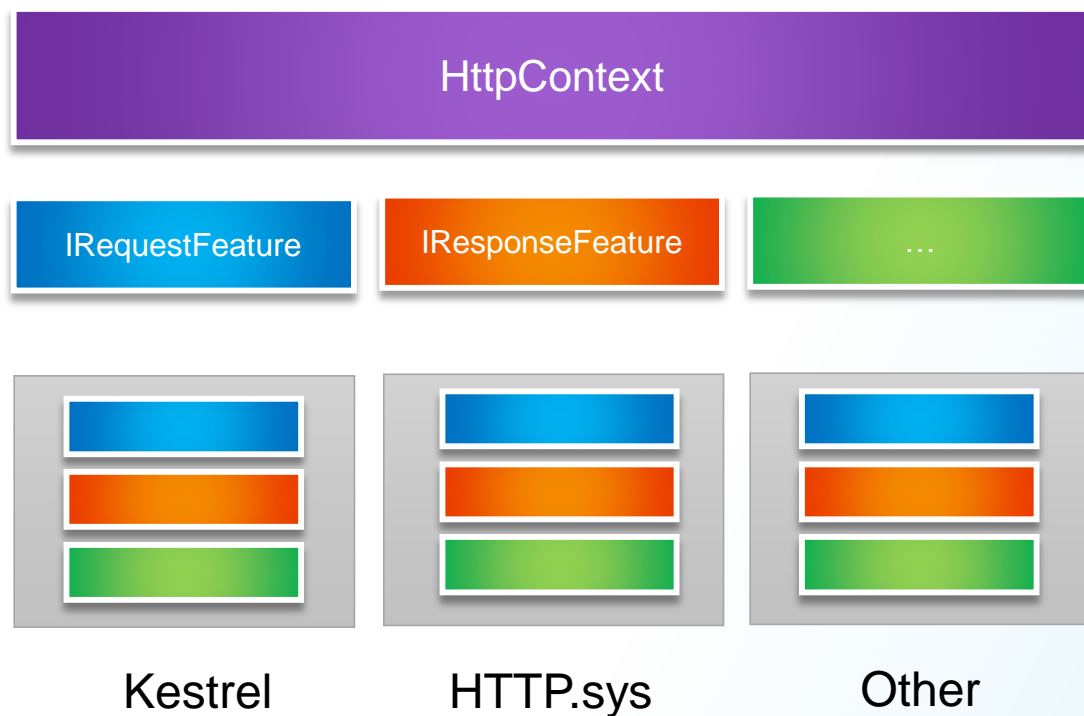
Server



Server + RequestDelegate

```
public interface IServer
{
    Task StartAsync(RequestDelegate handler);
}
```

Features



```
public interface IFeatureCollection
    : IDictionary<Type, object> { }
public class FeatureCollection
    : Dictionary<Type, object>,
      IFeatureCollection {}
public static partial class Extensions
{
    public static T Get<T>(
        this IFeatureCollection features);
    public static IFeatureCollection Set<T>(
        this IFeatureCollection features,
        T feature);
}
```

HttpRequest



```
public interface IHttpRequestFeature
{
    Uri Url { get; }
    NameValueCollection Headers { get; }
    Stream Body { get; }
}
```

```
public class HttpRequest
{
    private readonly IHttpRequestFeature _feature;
    public Uri Url => _feature.Url;
    public NameValueCollection Headers => _feature.Headers;
    public Stream Body => _feature.Body;
    public HttpRequest(IFeatureCollection features)
        => _feature = features.Get<IHttpRequestFeature>();
}
```

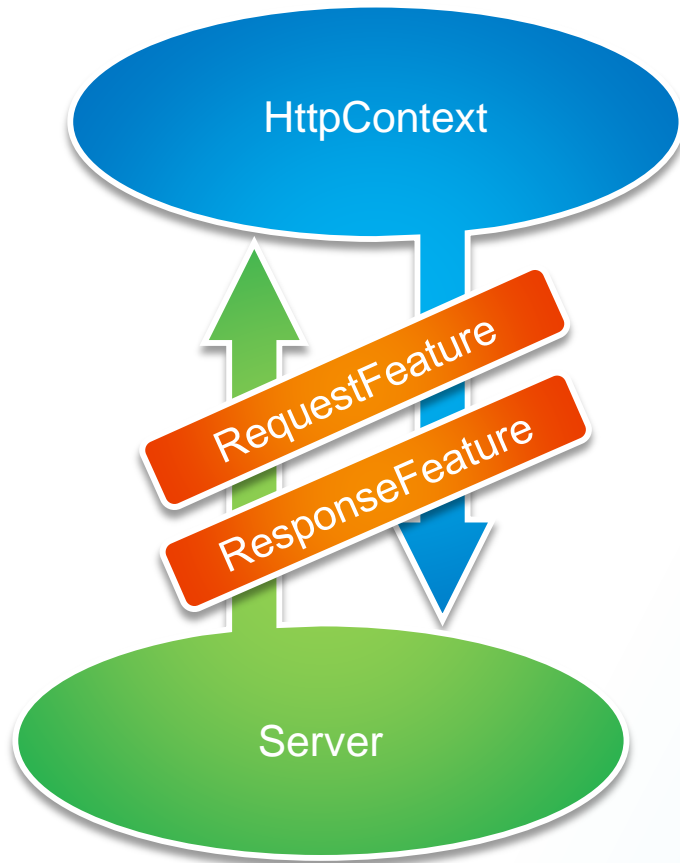

HttpResponse



```
public interface IHttpResponseFeature
{
    int StatusCode { get; set; }
    NameValueCollection Headers { get; }
    Stream Body { get; }
}
```

```
public class HttpResponse
{
    private readonly IHttpResponseFeature _feature;
    public HttpResponse(IFeatureCollection features)
        => _feature = features.Get<IHttpResponseFeature>();
    public NameValueCollection Headers
        => _feature.Headers;
    public Stream Body => _feature.Body;
    public int StatusCode {
        get => _feature.StatusCode;
        set => _feature.StatusCode = value; }
}
```

HttpContext



```
public class HttpContext
{
    public HttpRequest Request { get; }
    public HttpResponse Response { get; }
    public HttpContext(IFeatureCollection features)
    {
        Request = new HttpRequest(features);
        Response = new HttpResponse(features);
    }
}
```

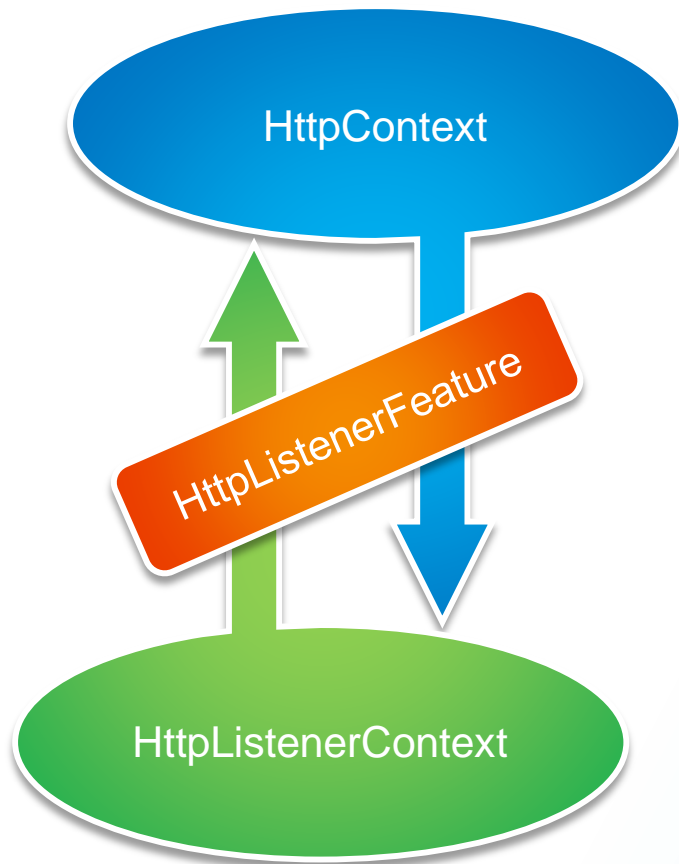
HttpListenerServer



```
public class HttpListenerServer : IServer
{
    private readonly HttpListener _httpListener;
    private readonly string[] _urls;
    public HttpListenerServer(
        params string[] urls)
    {
        _httpListener = new HttpListener();
        _urls = urls.Any()
            ? urls:
            new string[] { "http://localhost:5000/" };
    }
}
```

```
public async Task StartAsync(RequestDelegate handler)
{
    Array.ForEach(_urls, url => _httpListener.Prefixes.Add(url));
    _httpListener.Start();
    while (true)
    {
        var listenerContext = await _httpListener.GetContextAsync();
        var feature = new HttpListenerFeature(listenerContext);
        var features = new FeatureCollection()
            .Set<IHttpRequestFeature>(feature)
            .Set<IHttpResponseFeature>(feature);
        var httpContext = new HttpContext(features);
        await handler(httpContext);
        listenerContext.Response.Close();
    }
}
```

HttpListenerFeature



```
public class HttpListenerFeature : IHttpRequestFeature, IHttpResponseFeature
{
    private readonly HttpListenerContext _context;
    public HttpListenerFeature(HttpListenerContext context)
        => _context = context;
    Uri IHttpRequestFeature.Url => _context.Request.Url;
    NameValueCollection IHttpRequestFeature.Headers
        => _context.Request.Headers;
    NameValueCollection IHttpResponseFeature.Headers
        => _context.Response.Headers;
    Stream IHttpRequestFeature.Body => _context.Request.InputStream;
    Stream IHttpResponseFeature.Body => _context.Response.OutputStream;
    int IHttpResponseFeature.StatusCode
    {
        get { return _context.Response.StatusCode; }
        set { _context.Response.StatusCode = value; }
    }
}
```

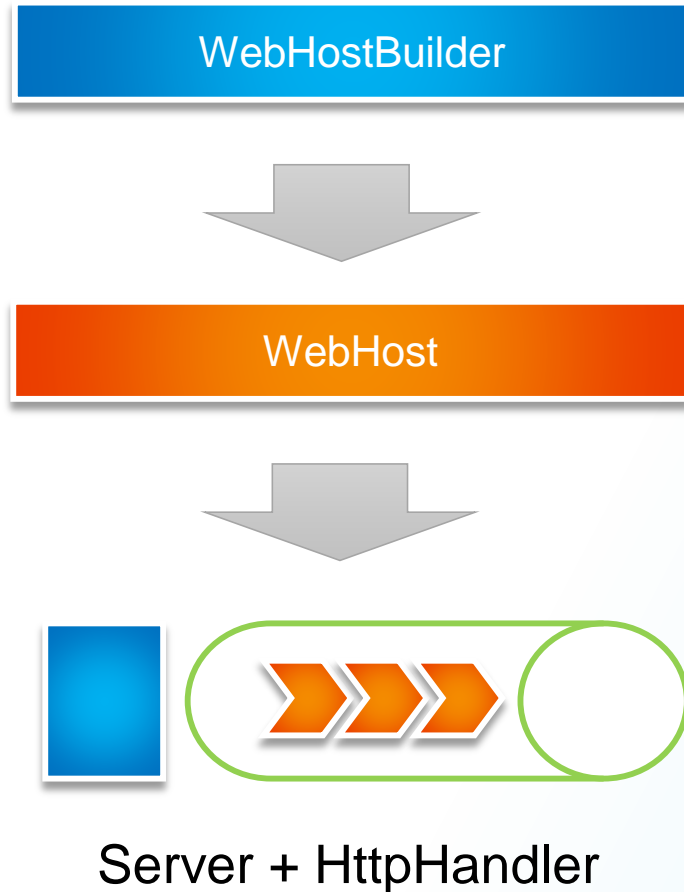
WebHost



Server + RequestDelegate

```
public interface IWebHost
{
    Task StartAsync();
}
public class WebHost : IWebHost
{
    private readonly IServer _server;
    private readonly RequestDelegate _handler;
    public WebHost(IServer server, RequestDelegate handler)
    {
        _server = server;
        _handler = handler;
    }
    public Task StartAsync() => _server.StartAsync(_handler);
}
```

WebHostBuilder



```
public interface IWebHostBuilder
{
    IWebHostBuilder UseServer(IServer server);
    IWebHostBuilder Configure(
        Action<IApplicationBuilder> configure);
    IWebHost Build();
}
```

WebHostBuilder



```
public class WebHostBuilder : IWebHostBuilder
{
    private IServer _server;
    private readonly List<Action<IApplicationBuilder>>
        _configures = new List<Action<IApplicationBuilder>>();

    public IWebHostBuilder Configure(
        Action<IApplicationBuilder> configure)
    {
        _configures.Add(configure);
        return this;
    }
    public IWebHostBuilder UseServer(IServer server)
    {
        _server = server;
        return this;
    }
}
```

```
public class WebHostBuilder : IWebHostBuilder
{
    public IWebHost Build()
    {
        var builder = new ApplicationBuilder();
        foreach (var configure in _configures)
        {
            configure(builder);
        }
        return new WebHost(
            _server, builder.Build());
    }
}
```

ASP.NET Core Mini



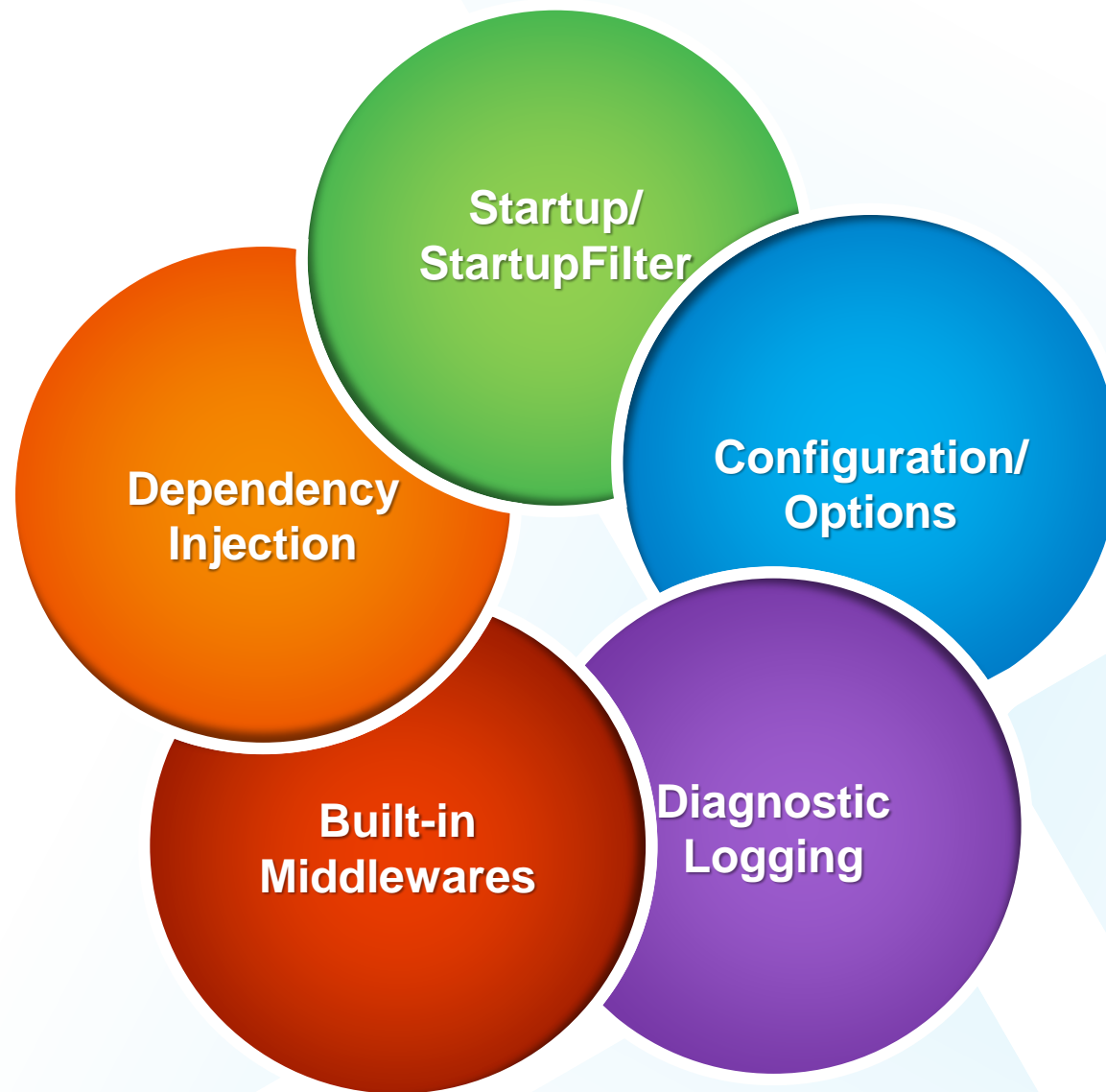
```
public class Program
{
    public static async Task Main()
    {
        await new WebHostBuilder()
            .UseHttpListener()
            .Configure(app => app
                .Use(FooMiddleware)
                .Use(BarMiddleware)
                .Use(BazMiddleware))
            .Build()
            .StartAsync();
    }
}
```

```
public class Program
{
    public static RequestDelegate FooMiddleware(RequestDelegate next)
    => async context => {
        await context.Response.WriteAsync("Foo=>");
        await next(context);
    };

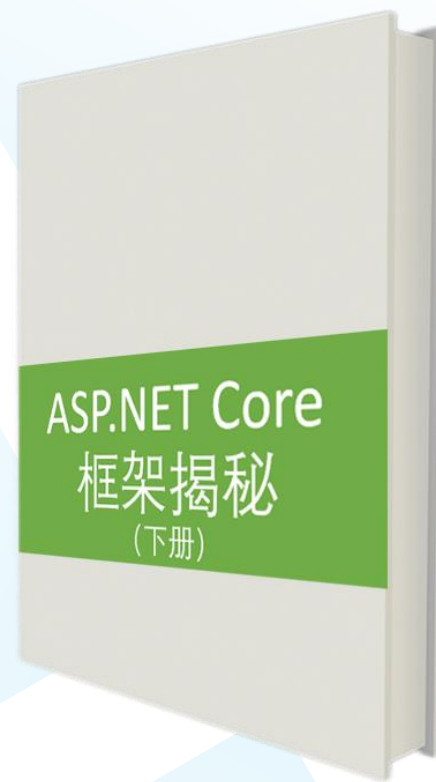
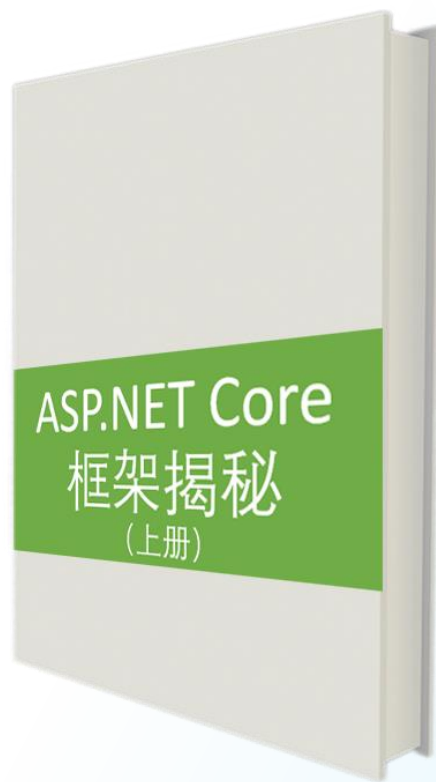
    public static RequestDelegate BarMiddleware(RequestDelegate next)
    => async context => {
        await context.Response.WriteAsync("Bar=>");
        await next(context);
    };

    public static RequestDelegate BazMiddleware(RequestDelegate next)
    => context => context.Response.WriteAsync("Baz");
}
```


漏掉了什么？



敬请期待...





Thanks

蒋金楠